Listing of claims:

1. (Currently amended) A computer-implemented method for processing a trace file of data accesses to obtain information that is used to improve memory usage for a computer program, comprising:

identifying repetitively occurring data access sequences and non-repetitively occurring data access sequences in the trace file; and

using the identified sequences to create a modified trace file, wherein the modified trace file includes data associated with the frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored. ~~by removing less frequently occurring data access sequences from the trace file~~.

2. (Original) The method of claim 1, wherein identifying the sequences includes steps, comprising:

constructing a grammar from the data accesses of the trace file;

building a candidate sequence using the grammar; and

if a cost of accessing data in the candidate sequence exceeds a threshold, marking the candidate sequence as a repetitively occurring data access sequence.

3. (Original) The method of claim 2, wherein computing the cost comprises multiplying a number of times the candidate sequence occurs in the grammar by a number of data accesses in the candidate sequence.

4. (Currently amended) The method of claim 1, further comprising using the identified data access sequences to update a stream flow graph that indicates the frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored. ~~how often each repetitively occurring data access pattern follows another repetitively occurring data access pattern.~~

5. (Original) The method of claim 1, wherein data accesses from the trace file are received as the computer program executes.

6. (Original) The method of claim 1, wherein the data access trace file is retrieved from a computer-readable medium.

7. (Previously presented) The method of claim 1, wherein the modified trace file is further processed to compress data in it by steps, comprising:

identifying other sequences of repetitively occurring data access sequences in the modified trace file; and

using the other sequences to create another trace by removing less frequently occurring data access sequences from the modified trace file.

8. (Original) The method of claim 7, wherein the other trace is used to pre-fetch data.

9. (Original) The method of claim 7, wherein the other trace is used in placing data in a cache.

10. (Currently amended) A computer-readable medium having computer-executable instructions encoded thereon for improving data accesses, the instructions comprising:

receiving data access information from an executing program;

~~identifying when the data access information is part of a frequently occurring data access pattern~~ identifying repetitively occurring data access patterns and non-repetitively occurring data access patterns; and

~~when the frequently occurring data access pattern follows another frequently occurring data access pattern,~~ updating a data structure to reflect the frequency that repetitively occurring data access patterns follow other repetitively occurring data access patterns when non-repetitively data access patterns are ignored. ~~that the data access pattern follows the other data access pattern.~~

11. (Original) The computer-readable medium of claim 10, wherein the data access information is received on a computer upon which the executing program is executing.

12. (Original) The computer-readable medium of claim 10, wherein the data access information is received on a computer other than a computer upon which the executing program is executing.

13. (Original) The computer-readable medium of claim 10, wherein a grammar representing the data access information is used in identifying when the data access information is part of a frequently occurring data access pattern.

14. (Original) The computer-readable medium of claim 10, wherein the data structure is a stream flow graph.

15. (Original) The computer-readable medium of claim 14, wherein the stream flow graph is used to pre-fetch data into memory.

16. (Original) The computer-readable medium of claim 15, wherein data is pre-fetched depending on the probability of the data being requested based on a current data access request.

17. (Previously presented) A computer-readable medium having a data structure stored thereon, comprising:

a database structured to store data access information that includes data access sequences of a computer program;

a stream flow graph structured to store data that indicates a frequency that a data access sequence follows another data access sequence; and

a pre-fetcher configured to use the data access information and the stream flow graph to fetch data elements into memory for use by the executing computer program.

18. (Previously presented) The computer-readable medium of claim 17, wherein the data structure further comprises timing information that is used to determine when the data element should be retrieved.

19. (Previously presented) The computer-readable medium of claim 17, wherein during requests for data in one data access sequence, pre-fetching begins for data in another data access sequence that will follow.

20. (Previously presented) The computer-readable medium of claim 19, wherein the other data access sequence follows when the one data access sequence dominates the other data access sequence.

21. (Original) A computer-readable medium having computer-executable components, comprising:

a database configured to store a stream flow graph;

a database configured to store data access sequence information; and

a cache memory manager coupled to the stream flow graph database and the data access sequence database, wherein the cache memory manager is configured to arrange data elements of a repetitively accessed data stream in a cache using information from the two databases.

22. (Original) The computer-readable medium of claim 21, wherein the data elements of one repetitively accessed data stream are arranged in the cache to avoid a cache conflict.